Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rihard Gaberšček

Model učenca v inteligentnih tutorskih sistemih

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: prof. dr. Saša Divjak

Ljubljana, 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo: Model učenca v inteligentnih tutorskih sistemih

Tematika naloge:

Proučite značilnosti modela učenca in raziščite obstoječe teorije v luči inteligentnih tutorskih sistemov. Predlagajte primerno rešitev in realizirajte ustrezno aplikacijo za mobilne naprave z operacijskim sistemom Android.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Rihard Gaberščček, z vpisno številko **63070081**, sem avtor diplomskega dela z naslovom:

Model učenca v inteligentnih tutorskih sistemih.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Saša Divjaka,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 2. aprila 2015 Podpis avtorja:

Rad bi se zahvalil družini za podporo in potrpežljivost med študijskimi leti in prijateljem za druženje in pomoč. Pravtako bi se zahvalil punci Vesni za vse nasvete in podporo.

Svoji dragi Vesni.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
BN	Bayesian networks	Bayesove verjetnostne mreže
ITS	Inteligent Tutor Systems	Inteligentni tutorski sistemi
SM	Student model	Model učenca
OCC	Ortony, Clore, and Collins theory	Teorija Ortony, Clore, and Collins
JSON	JavaScript Object Notation	Objektna notacija JavaScript
ADT	Android Development Tools	Razvojna orodja Androida

Povzetek

Cilj diplomskega dela je teoretično raziskovanje modela učenca v inteligentnih tutorskih sistemih (ITS) in praktična realizacija konceptov na operacijskem sistemu Android, glede na hitro rast mobilneg segmenta in pomanjkanje ITS-jev na mobilnem področju. Predstavljeni so različni pristopi izdelave modela učenca in učnih stilov. Drugi del naloge predstavlja implementacija preprostega ITS-ja, kjer je model učenca zgrajen s pomočjo mehke logike, stereotipov in prekrivnega modela. Učno domeno v našem primeru predstavljajo osnovne operacije z ulomki. Aplikacija podaja personalizirano vsebino glede na trenutno stanje modela učenca. Učencu prav tako nudi personalizirane pomoči in vrača povratne informacije za optimalnejše učenje.

Ključne besede: model učenca, ITS (inteligentni tutorski sistemi), model mehke logike, Android.

Abstract

The aim of this thesis is a theoretical research of the student model in intelligent tutoring systems (ITS) and a practical realisation of the concepts on Android OS, due to the rapid growth of the mobile segment and a lack of ITS in the mobile field. Various approaches in creating a student model and learning styles are presented. The second part of the thesis introduces the implementation of a simple ITS, where the student model is built using fuzzy logic, stereotypes and an overlay model. In the present case, the learning domain is represented by basic operations with fractions. The application provides personalised content based on the current state of the learner model. In addition, it offers personalised assistance, and a feedback for optimal learning.

Keywords: Student model, ITS (intelligent tutoring systems), fuzzy model, Android.

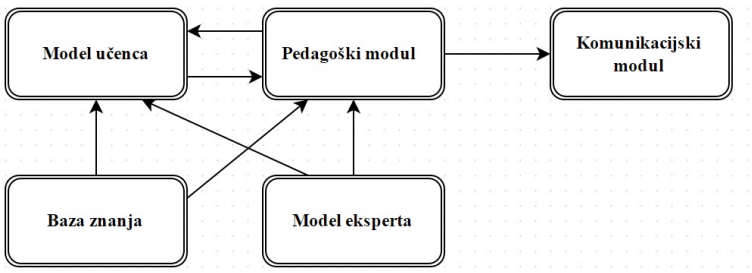
Poglavje 1

Uvod

Ena poglavitnih lastnosti, ki nam je omogočila, da smo kot vrsta tako uspešni, je znanje in njegovo širjenje. Zaradi naše socialne naravnanosti velikokrat z opazovanjem povzamemo stvari, ki so se jih naučili drugi. Vendar je aktivno, strukturirano deljenje znanja veliko bolj učinkovito. S tem dobimo učence in tutorje, izobražence, ki so specializirani za optimalno širjenje znanja. Najbolj optimalen način prenos znanja je individualno tutorstvo, kjer se lahko tutor v celoti posveti učencu in ga tudi celostno pozna. Seveda pa takšni načini niso primerni za skaliranje na večje modele, zato je moderno poučevanje bazirano na tutorju in omejeni skupini učencev. Ne glede na majhnost skupine so razlike z individualnim tutorstvom opazne.

ITS-ji imajo korenine v mehanskih aparatih, ki so pomagali učencem beležiti pravilne odgovore, formalizirati snov, itd... Nadaljevalo se je s sistemi, ki so na podlagi učenčevega odgovora vodili reševanje (v smislu odgovor ena ima posledico ena, odgovor dva ima posledico dva). Takšni sistemi niso razlikovali med posameznimi učenci.

ITS-ji so računalniški sistemi, ki želijo simulirati individualno poučevanje. Učne vsebine, namige, povratne informacije in vse ostale elemente hočemo prilagajati na način, da jih bo učenec čim lažje in čim hitreje osvojil. ITS lahko idejno razdelimo v štiri komponente, ki so lahko še naprej deljive in med seboj povezane. Kot lahko vidimo na sliki 1.1, so to: model učenca, baza znanja, model eksperta (v nekaterih literaturah namesto model eksperta, navajajo uporabniški vmesnik) in pedagoški modul.



Slika 1.1: Prikaz splošnih komponent ITS-ja in interakcije med njimi.

Model učenca je entiteta v sistemu, kjer hranimo podatke o individualnem uporabniku sistema. Zaradi omejenega komunikacijskega kanala med učencem in sistemom, so takšni podatki lahko nepopolni ali delno napačni. Hranimo podatke, kot so na primer napredek pri reševanju problema, lastnosti, učni stili, različne statistike, napačno naučena dejstva itd... Na podlagi teh podatkov potem s pomočjo pedagoškega modula serviramo učencu nove učne probleme, vendar moramo biti pozorni na omejitve.

Pedagoški modul naj bi nadomestil tutorja in se sam odločal, kdaj je učenec zadovoljivo osvojil znanje in tudi pripravljal akcije na podlagi SM in pretekle interakcije. Primer bi bil, zamenjava učne teme, oziroma prehod na novo poglavje. Modul mora pregledati dosedanjo interakcijo in na podlagi zbranih podatkov določiti temo, ki jo bo kot naslednjo obdelal učenec, kakšna bo težavnost in učna strategija. Prav tako mora skrbeti da je učenec dovolj motiviran in da napreduje. Skrbi tudi za prikaz namigov, ki jih lahko glede na preteklo učenčevo uspešnost ponujamo v različnih subtilnostih. Upoštevati mora še utrjevanje in ponavljanje snovi. Vendar obstajajo tudi pasti, kot so, na primer, vsiljivi tutorji, ko pedagoški modul deluje na nepopolnem SM in dajemo vsiljive namige ali pa neprimerno težavnost.

Baza znanja predstavlja nekakšen virtualni učni načrt. Hrani vsa dejstva, ki jih takšen sistem vsebuje in poučuje. Biti mora odprta, da lahko tudi drugi moduli dostopajo do nje. Velik izziv predstavljata razširljivost in dodelave. Pojavi se tudi vprašanje, kako naj bodo podatki predstavljeni, kot dejstva ali kot mentalni model oziroma koncept. Zato je izgradnja baze znanja, ki bi upoštevala vsa načela, precejšen izziv.

Model eksperta (ta modul ni obvezen in se ne pojavlja v vseh ITS-jih) pred-

stavlja postopke in načine reševanja nalog, kot bi jih izvajal nekdo, ki poučevano snov že obvlada.

Inteligentni tutorski sistemi se ponavadi ne uporabljajo pri učnem procesu. Glavnina dosedanjih raziskovalnih naporov se je usmerjala predvsem na osebne računalnike, kljub velikemu potencialu pa je ta koncept zapostavljen na področju mobilnih platform. V diplomskem delu skušamo podati koncepte, ki bi omogčili zasnovo in realizacijo ITS na mobilni platformi. Interakcija med uporabnikom in učno snovjo, ki jo uporabnik upravlja z dotikom, je bolj naravna in podobna klasičnim metodam kot na osebnem računalniku.

V nadaljevanju diplomskega dela se bomo osredotočili predvsem na model učenca in načine modeliranja le tega. Podali bomo rešitev za mobilno platformo Android z uporabo knjižnice jfuzzylite. V ta namen smo sprogramirali preprost ITS, kjer je model učenca predstavljen kot kombininacija prekrivnega, stereotipnega in modela mehke logike.

Poglavje 2

Model učenca

Model učenca je v ITS-ju eden najpomembnejših modulov, saj hrani informacije o učencu. Glavni namen ITS-ja je učinkovita podpora učenja, za kar moramo dobro poznati uporabnika, oziroma moramo prepoznati ključne lastnosti, ki nam to poznavanje omogočajo. Obstaja veliko različnih modelov, ki so se izkazali za obetajoče na specifičnih področjih, vendar noben ni dovolj splošen, da bi omogočal prenosljivost na druge sisteme brez ponovne implementacije. Vsekakor pa velja konsenz, da je izgradnja takšnega modela zaenkrat ročno in dolgotrajno opravilo. Ponavadi terja sodelovanje strokovnjakov z več področij, metode za avtomatsko izdelavo pa so še v začetnih fazah in omejenem obsegu. Poglejmo nekatere splošne karekteristike takega modela.

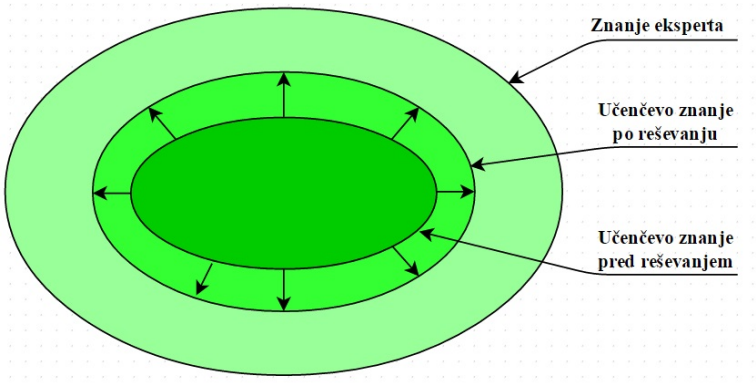
2.1 Modelirane karakteristike

Izbira modela je učinkovita, kadar lahko modeliramo domensko odvisne in neodvisne parametre. Modelirani parametri so lahko statični ali, nestatični / spremenljivi. Statični so: ime, priimek, starost in podobni ne spremenljivi podatki, ki jih lahko zajamemo z vprašalniki. Spremenljivi parametri pa se spreminjajo med reševanjem nalog in so odvisni od znanja (obstoječe, naučeno) in sposobnosti, učnih stilov in preferenc, napačno naučenih pravil, čustvenih in kognitivnih ter metakognitivnih faktorjev. Prav tako so karekteristike odvisne od izbire modela. Na primer, v nekaterih kognitivnih modelih skušamo zajeti učenčeva čustva, drugi modeli pa so usmerjeni v zajem parametrov znanja, okolja itd... Za težavne so

se izkazali tudi modeli, ki so bili preveč podrobni, kar zelo oteži opravila drugih modulov v ITS-ju. Kompleksnost pedagoškega modula, se veča s kompleksnostjo modula učenca. Vendar to ne pomeni, da v SM ne smemo vključiti lastnosti, ki jih pedagoški modul ne uporablja, le da so te namenjene izključno raziskovalnemu delu. Poglejmo nekaj najpogosteje uporabljenih modelov v zadnjih letih.

2.2 Prekrivni model

Pri tem modelu sklepamo, da ima učenec nepopolno a pravilno predznanje o učni domeni (vse učno gradivo v sistemu), kot prikazuje slika 2.1. Je podmodel domenskega modela. Deluje tako, da učno domeno razdelimo na segmente in na podlagi poznavanja segmentov ocenimo poznavanje domene. Primeren je za sisteme, kjer imamo učno snov razdeljeno v hierarhijo. Slabost tega modela je, da se učenec ne bo naučil tematik, ki jih ekspert(tisti, ki je sistem postavil) ni uvrstil v model. Odsoten je tudi sistem za zajemanje napak (t. i. "bug library"). Takšen model je primeren za ocenjevanja znanja domene.



Slika 2.1: Skica prekrivnega modela. Večji krog predstavlja znanje sistema, manjši pa učenčevo osvojeno znanje.

Primeri programov, ki uporabljajo prekrivni model: MEDEA, INFOMAP, ICICLE, PDINAMET, GUIDON.

2.3 Stereotipni model

Pri izdelavi stereotipnega modela učence razvrstimo na podlagi podobnih lastnosti, kot to določa izrek 2.1. Če učenec ima lastnost (tMi), ga uvrstimo v stereotip, če pa ima lastnost (rMj), pa ga iz skupine odstranimo. Ta oblika modeliranja se po navadi uporablja v kombinacijami z drugimi modelirnimi tehnikami. Primerna je za zajem učnih stilov, preferenc in ostalih kognitivnih sposobnosti (pozornosti, dojemanja in spomina).

Izrek 2.1 *Enačba za klasifikacijo, kjer so tMi pogoji sprožitve z binarno vrednostjo true false, rMj pa so pogoji odstranitve iz stereotipa.*

$$\exists i, tMi = true \rightarrow active(Mg)\ ali \tag{2.1}$$

$$\exists j, rMj = true \rightarrow not\ active(Mg) \tag{2.2}$$

Primeri programov, ki uporabljajo stereotipni model: INSPIRE, WELSA, AUTO-COLLEAGUE, CLT.

2.4 Pertubacijski model

Pertubacijski model je razširitev prekrivnega modela, ki vključuje pedagoško pomembne učenčeve napake, kjer še gradimo sistem za zajemanje napak (”bug-library”). Na ta način lahko profiliramo in odpravljamo napačno naučene vzorce. Sistem poskuša z eleminacijo izluščiti, katera učna osnova je napačno naučena. Bug library je lahko v naprej zgrajen ali pa ga gradimo med samo analizo učenca. Takšno modeliranje je uporabno predvsem za zajem napačno naučenih snovi.

Primeri programov, ki uporabljajo pertubacijski model: LeCo-EAD, InfoMap.

2.5 Strojno učenje

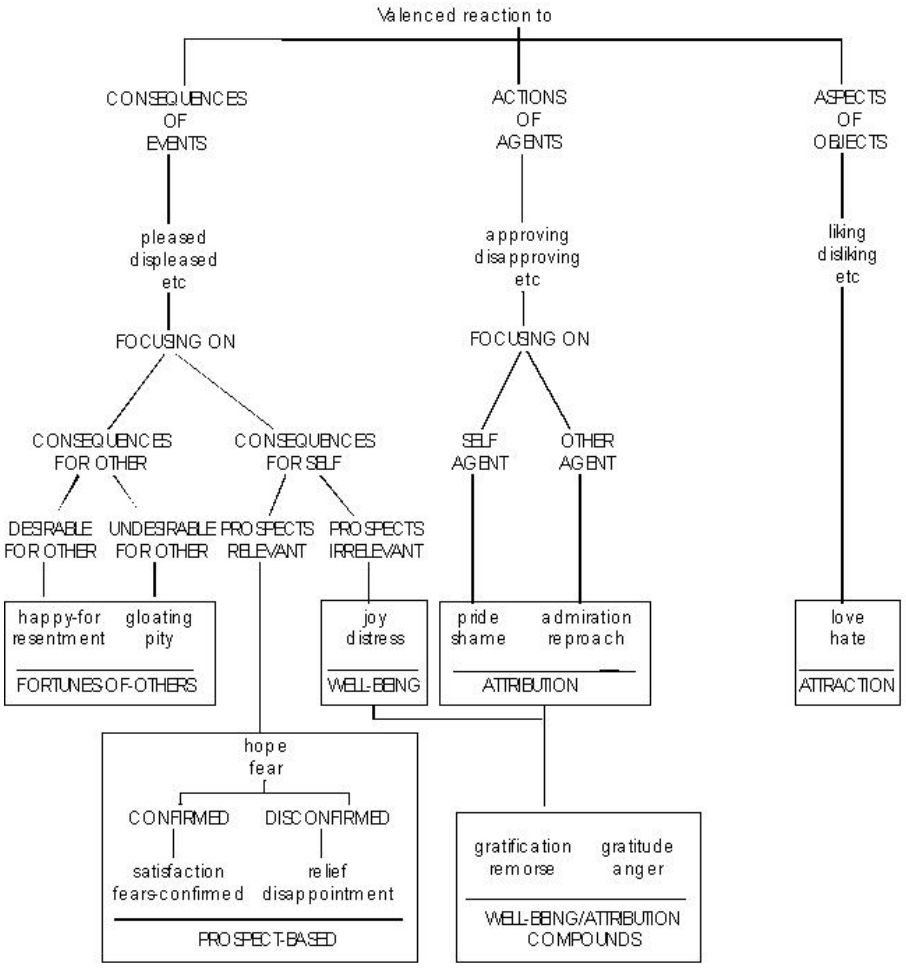
Strojno učenje se uporablja predvsem za izgradnjo učnega modela iz opazovanih akcij učenca in izgradnjo oziroma razširitev sistema za zajemanje napak. Prevladujeta predvsem Bayesove verjetnostne mreže in model mehke logike, ki si ju bomo podrobneje ogledali v nadaljevanju.

2.6 Kognitivni modeli

Kognitivna psihologija je veda, ki poskuša dognati in formalizirati procese, kot so pogojno učenje, percepcija, spomin, reševanje problemov, pozornost itd. Ker je model učenca usmerjen v zagotavljanje čim večjega učnega izkoristka, so mnogi raziskovalci predlagali modele, ki temeljijo na kognitivni psihologiji. Razvoj takšnega modela zahteva koordinacijo strokovnjakov s področja domene, kognitivne psihologije in programerjev. Kognitivni model sestavljajo pravila in sposobnosti, ki simulirajo kako, učenec pristopi k problemu. Podrobneje bomo opisali model, ki temelji na teoriji Ortony, Clore, and Collins.

2.6.1 Teorija Ortony, Clore, and Collins

Teorija je bila razvita za strojni zajem in strojno ugotavljanje čustev. Kot prikazuje slika 2.2 temelji na treh vidikih: dogodki, agenti in objekti. Nekdo je lahko zadovoljen / nezadovoljen s posledicami dogodka, lahko sprejema / zavrača akcije agenta in lahko sprejema / zavrača objekt. Nadaljnja delitev je, da imajo lahko dogodki posledice zase ali za druge in da je lahko agent druga oseba ali pa učenec. Posledice dogodka delimo na želene / neželene, posledice za učečega pa na relevantne / nerelevantne. Relevantne posledice delimo na izvedene / neizvedene. Vsem tem spremenjlivkam določimo vrednost in utež ter prag, pod katerim čustvo ni več relevantno. Pozitivna in negativna čustva so odvisna od željenosti dogodka v povezavi s cilji, primer so negativna čustva v izreku 2.2. Model OCC predpostavlja, da so čustva pozitivna, če se zgodi želeni dogodek, in negativna, če se zgodi neželeni dogodek. Primer negativnih čustev je, ko zagret učenec dobi slabo oceno. Gradnja sistema je imela tri stopnje: klasifikacijo mogočih dogodkov v sistemu, zajem učenčevih ciljev (podlaga za učenčevo klasifikacijo) in klasifikacija dogodkov na želene / neželeene. V opisanem sistemu so za zajem ciljev uporabili vprašalnik (MSLQ). Na utež čustev vpliva predvsem stopnja pričakovanosti, ker imamo v sistemu tudi nepričakovane dogodke. Primer: ko povprečen učenec doseže izjemen rezultat, ima veselje večjo utež, kot pri učencu ki je takih rezultatov vajen in jih pričakuje. Učenčeva čustva pri reševanju nalog nato pedagoški modul uporabi kot izhodišča za prikaz vsebine.



Slika 2.2: Shema teoretičnega modela OCC. [16]

Izrek 2.2 *Primer pravila v prologu, ki predstavlja nezadovoljstvo učenca z izidom reševanja. [14]*

```
bel (ag, event_pleasantness(not_correct_answer,
displeased)) if
bel (ag, student_goal(performance)),
bel (ag, event(not_correct_answer)).
/* It is a prospect of an event */
bel (ag, is_prospect_event(not_correct_answer)) if
bel (ag, event(not_correct_answer)).
```

```
/* When the student is displeased, disappointment and
distress emotions arise */
bel (ag, student_emotion(disappointment)) if
bel (ag,event_pleasantness(Event,displeased)),
bel (ag,-is_mediador_action),
bel (ag,is_prospect_event (Event)).
bel (ag, student_emotion(distress)) if
bel (ag, event_pleasantness(Event,displeased)),
bel (ag, -is_mediador_action).
```

Primeri teorij, ki so bile uporabljene za izgradnjo SM, so: Human Plausible Reasoning (HPR) theory, the Multiple Attribute Decision Making (MADM) theory, the Ortony, Clore, and Collins(OCC) teorija in kontrolno-vrednostna teorija.

Primeri programov, ki uporabljajo kognitivni model: F-SMILE, Web-IT, VIRGE, AMPLIA, Olympia. Architecture.

2.7 Omejujoči model

V tem modelu so napake in učenčevo znanje predstavljene kot omejitve. Model temelji na prepričanju, da čeprav se je učenec pravilno naučil zahtevano snov, med reševanjem naloge še vedno prihaja do napak. Omejitev je predstavljena kot utež ustreznosti in utež zadovoljstva. Utež ustreznosti mora veljati, da je omejitev veljavna za dano nalogo, nakar preverimo še utež zadovoljstva, ki tudi mora veljati, da je rešitev pravilna. Primer omejitve v ITS-jih, ki preverjajo znanje programiranja, je na primer sintaksa, semantika in stil učenčeve rešitve. Slika 2.3 prikazuje primer semantične omejitve. V primeru, da utež ustreznosti velja in utež zadovoljstva ne velja, potem rešitev klasificiramo kot "bug"oziroma napako. Domena znanja je predstavljena kot set omejitev, model učenca pa je predstavljen, kot množica prekršenih omejitev. Model zaradi svoje zasnove omogoča tudi dajanje namigov, ki temeljijo na prekršenih omejitvah. V zadnjem času se tudi pojavljajo rešitve, ki z različnimi tehnikami poskušajo z avtomatsko izgradnjo omejitev.

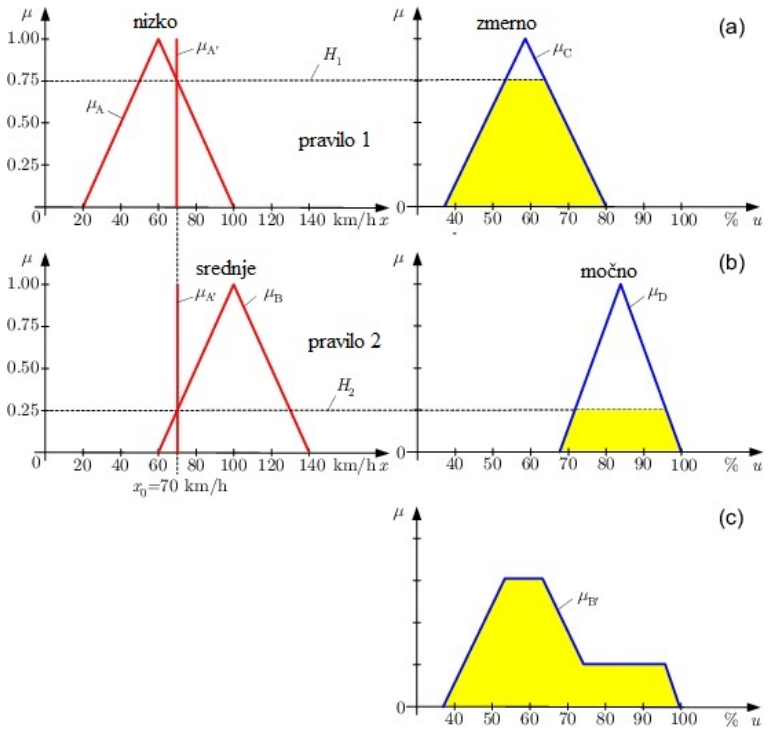
Izrek 2.3 *Primer semantične omejitve v SQL-Tutorju, ki preverja, če so rezultati urejeni po padajočem vrstnem redu.[8]*

```
(p 361
 "Check whether you should have ascending or descending order!"
(and (not (null (order-by is)))(not (null (order-by ss)))
(bind-all '?n (names (order-by ss)) bindings)
(match '(?*d1 ?n "DESC" ?*d2) (order-by ss) bindings)
(not (qualified-name '?n)))
(match '(?*d3 ?n "DESC" ?*d4) (order-by is) bindings)
"ORDER BY")
```

Primeren je predvsem za zajem učenčevega znanja. Primeri programov, ki uporabljajo stereotipni model: J-LATTE, CAS, SQL-Tutor, COLLECT-UML.

2.8 Model mehke logike

Mehka logika, je razširitev teorije množic, saj je lahko element član, nečlan ali delni član množice 2.4. Zaradi nedeterminističnih elementov in nepopolnih podatkov so raziskovalci razvili model mehke logike (Fuzzy model), ki se bolje spopada z negotovostjo v sistemu. Prednost tega modela je njegova zmožnost delovanja z besedami, kar pomeni, da so lahko ocene in razredi opisni. Model je sestavljen iz štirih korakov: mehčanje (fuzzification), agregacija, kompozicija in odmehčanje (defuzzification). V prvem koraku glede na vhodne parametre izračunamo stopnjo pripadnosti in jo preslikamo v besedno oceno, kot prikazuje slika 2.3. V koraku agregacije za vsako lastnost na podlagi stopnje pripadnosti izračunamo utež veljavnosti, nakar lahko vzamemo minimalno vrednost ali produktni nabor vrednosti, odvisno od izbire tipa modela. S kompozicijo izračuna vrednost vseh besednih ocen istega razreda (izračunamo utež korelacij med vrednostmi) ugotovimo utež razreda, združimo jih z maksimalno (max) največjo funkcijo ali vsoto. Odmehčanje (defuzzification) vključuje preslikavo besedne ocene in njihovih uteži nazaj v številčno oceno, ponavadi z metodo center maksimuma (Centre of Maximum). Primeren je predvsem za zajem čustev , kognitivnih in meta-kognitivnih funkcij.



Slika 2.3: Primer mehkega računanja, (a) Pri vhodni vrednosti 70 km/h spremennljivke nizko, izračunamo pripadnost izhodne spremenljivke zmerno.(b) Pri vhodni vrednosti 70 km/h spremennljivke srednje, izračunamo pripadnost izhodne spremenljivke močno. (c) sestavljen lik, ki predstavlja izhodno mehko množico, unijo množic zmerno in močno. [7]

Izrek 2.4 *Model lahko definiramo kot urejeno množico (x, $\mu_A(x)$), kjer je $x \in X$ in $\mu_A(x) \in [0, 1]$, $\mu_A(x)$:X→ [0, 1] pa je funkcija pripadnosti. Vrednosti $\mu_A(x)$ pravimo tudi mera pripadnosti.*

$$\mu_X = \begin{cases} 1, \ x \ absolutno \ v \ A \\ 0, x \ absolutno \ ne \ v \ A \\ (0,1), \ x \ delno \ v \ A \end{cases} \tag{2.3}$$

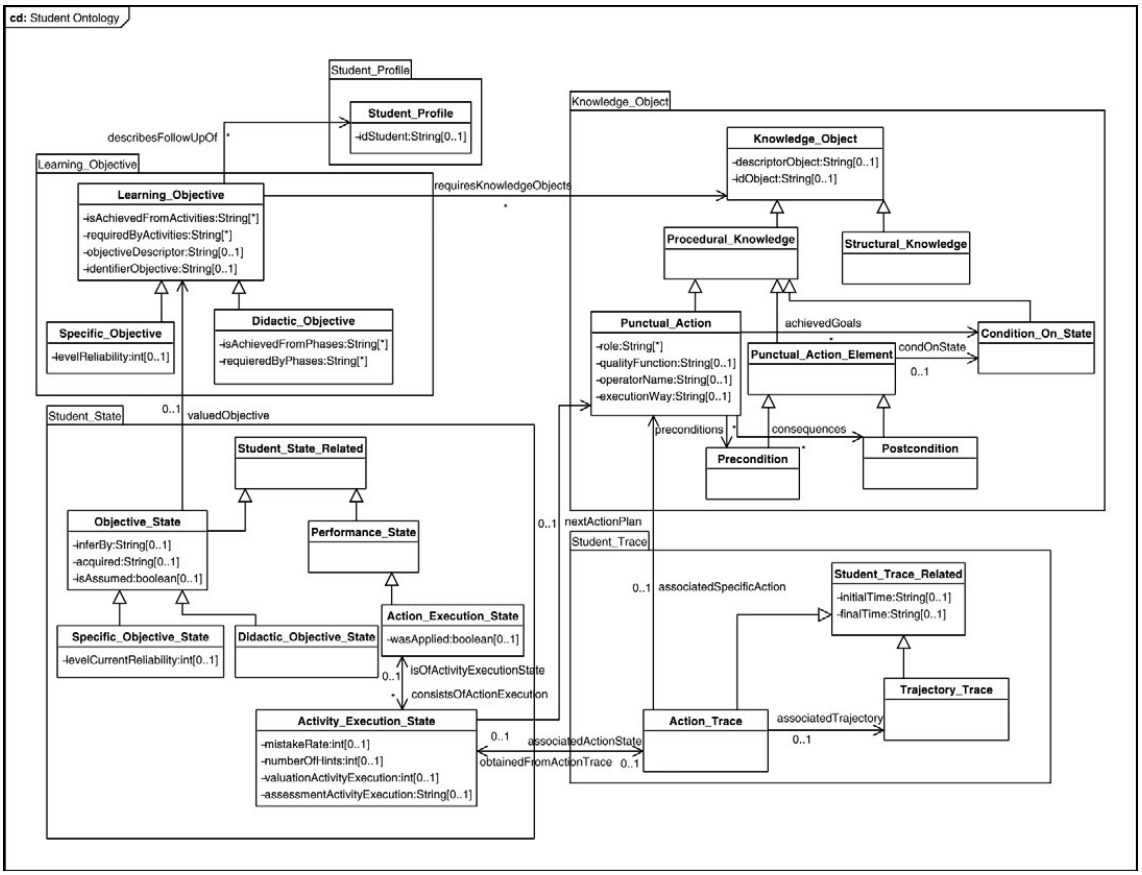
2.9 Bayesove verjetnostne mreže

Bayesove mreže so usmerjeni, aciklični graf, kjer vozlišča predstavljajo spremenljivke, povezave pa verjetnostno odvisnost med spremeljivkami. Vozlišča v učnem modelu predstavljajo komponente učenca (znanje, napačno naučene koncepte, čustva, učne stile, motivacije, cilje ...). Imamo tri različne modele v odvisnosti od izdelave:
-ekspertno usmerjeni model
-učinkovitostno usmerjeni model
-podatkovno usmerjeni model
Uporabljajo se predvsem za zajem čustev in občutkov učenca in metakognitivne lastnosti, kot so na primer samoregulacija, samokritičnost itd.

Primeri programov, ki uporabljajo Bayesove mreže: SMILE, ANDES, ACE (Adaptive Coach for Exploration), English ABLE, AMPLIA.

2.10 Ontološki model

Termin ontologija označuje računalniško berljive metapodatke o temi. Na tak način hočemo zagotoviti računalniško prepoznavanje vsebine. Celotno področje teži k izgradnji nekakšne digitalne knjižnice znanj, ki so lahko razumljive za strojno obdelavo. Ontološki model učenca, teži k prenosljivosti in ponovni uporabnosti, medtem ko so drugi modeli predvsem specifični za aplikacijo, za katero so bili zasnovani. Realizacija modela se je zaradi standardov in podprtosti usmerila predvsem na Web Ontology Language. Model naj bi bil sestavljen iz dveh glavnih enot: splošnih podatkov in interakcije s sistemom. Groba skica takega sistema je vidna na sliki 2.4. Splošni podatki o učencu naj bi vsebovali predvsem statične podatke, kot so učni cilji, učni stili, podatki o učencu (ime, priimek, mail itd). Interakcija s sistemom ali obnašanje učenca opisuje učenčev merljiv odziv na stimulacijo v sistemu. Sem shranjujemo učenčevo poznavanje domene, njegovo uspešnost, čas reševanja naloge in podobne parametre.



Slika 2.4: Primer ontološkega modela v sistemu IVETs [11]

Primeri programov, ki uporabljajo Ontology model: OPAL(framework), MA-EVIF, SONITS, ELENA project, IVETs.

2.11 Primerjava modelov

Najbolj razširjena modela zadnjih nekaj let sta prekrivni in stereotipni model. Zaradi nedeterminističnosti se je vedno bolj uveljavljal tudi model mehke logike, ki pa ga v zadnjih letih nadomešča probabilistični model (BN). V zadnjih letih se veča tudi zanimanje za ontološki model. Najbolj splošno oceno pa podajajo hibridni modeli. Ti združujejo več osnovnih, ki modelirajo različne karekteristike in nam omogočajo celovitejše ocene učenca. Prevladujejo kombinacije dveh modelov,

poskusov kombiniranja večih pa zaradi kopleksnosti ni veliko. Najpogostejše kombinacije treh sistemov predstavljajo prekrivni model, stereotipni model in model mehke logike.

2.12 Učni stili

Učni stil je termin, ki združuje več teorij na področju klasifikacije posameznikovih preferenc pri oblikovanju znanja. Teorija učnih stilov se ukvarja s tem, kako nova znanja najlažje predstaviti posamezniku, tako da so njemu naravno najbližje. Obstaja več ne nujno skladnih teorij in še več različnih opredelitev, podrobneje bo predstavljena samo metoda učnih stilov.

Pri metodi učnih stilov učence klasificiramo v tri kategorije (vizualni, avditorni in kinestetični), glede na preferiran način zajemanja znanja.

Vizualni učenci najhitreje osvojijo novo znanje, predstavljeno v obliki slik, grafov, preglednic, knjig itd... Pri učenju si radi pomagajo z vizualizacijo, imajo dobro prostorsko predstavo, bolje dojemajo oblike, barve, velikosti, oblike itd... Bolj so pozorni na telesno govorico. Takšni učenci naj bi pri učenju uporabljali več vizualnih pripomočkov in barvanje pomembnega besedila. Pomaga jim tudi, da vidijo učitelja, saj s tem zaznavajo telesno govorico. Miselni vzorci so tipično orodje, na katerega se vizualni učenci ponavadi zanesejo.

Auditorni učenci so najbolj dovzetni za zvočne dražljaje: predavanja in razprave so tipični primeri. Lažje se spomnijo slišanih informacij, imajo boljše govorne sposobnosti (širše besedišče) in so bolj glasbeno nadarjeni (lažje ločijo tone, ritem). Priporočila za auditorne učence so, naj sodelujejo v učnih razpravah(med sošolci in tutorji), naj snov ponavljajo na glas, naj raje snemajo zapiske (namesto zapisovanja) in uporabljajo na primer avdio e-gradiva.

Kinestetični učenci se najbolje odzovejo na stvari, ki jih lahko otipajo, in na telesne gibe. Takšni učenci imajo radi praktično poučevanje in postanejo nemirni, ko so dlje časa na enem mestu. Dobro se tudi vživijo v igranje vlog in na stvari gledajo širše. Za takšne učence se predlaga pogosto jemanje odmorov med učenjem in veliko gibanja, naj najprej preletijo celotno poglavje, naj si ustvarijo grobo sliko in učenje v skupini.

Poglavje 3

Zasnovan sistem

3.1 Na kratko o aplikaciji

V teoretičnem delu diplomskega dela smo se osredotočili na model učenca v inteligentnih tutorskih sistemih. V praktičnem delu bomo opisali zasnovan sistem, ki se trudi teoretična spoznanja prikazati v praksi. Cilj praktičnega dela je bil izdelava sistema, ki na podlagi trenutnega modela učenca prikazuje personalizirano vsebino.

Izdelali smo aplikacijo za operacijski sistem Android. Za realizacijo modela učnenca smo se odločili za kombiniran model stereotipnega, prekrivneg in modela mehke logike. Stereotipni model smo vzeli predvsem za podlago, za zajem fiksnih parametrov, kot so podatki in učni stil. Izgradnja modela poteka preko vprašalnika, kjer učenec vnese zahtevane podatke kot so ime, priimek, starost, spol ... Za klasifikacijo učnih stilov uporabljamo metodo iz "the motivated strategies for learning questionnaire"(MSLQ) [21].

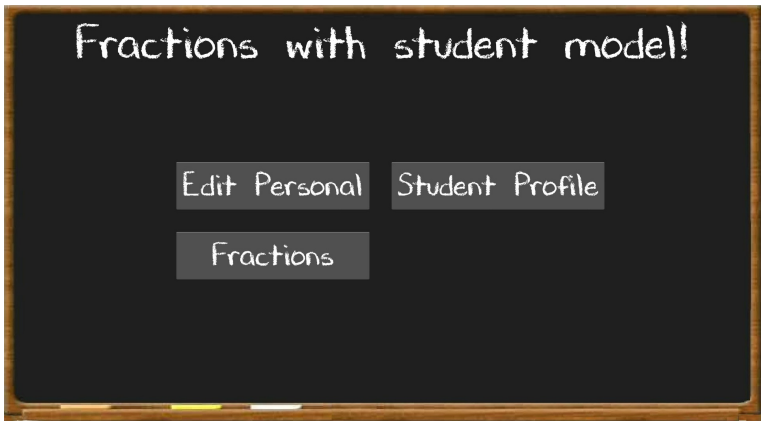
Dinamične lastnosti zajemamo med reševanjem in se nanašajo predvsem na uspešnost. Učno domeno smo omejili na osnovne operacije z ulomki in jo razdelili na poglavja (primerjava ulomkov, seštevanje, odštevanje, množenje in deljenje). Vsako poglavje, razen primerjave, ima tri težavnostne stopnje, ki se razlikujejo po tem, kje se nahaja neznanka. Poimenovali smo jih lahka, srednja in težka. Prva ima neznanko na strani rezultata, ostali dve pa nekje med operatorji, tako da je na višjih stopnjah za reševanje nalog treba uporabljati tudi obratno logiko.

Primarna strategija za spoznavanje in utrjevanje matematičnih nalog je ponavljanje. Zato v načinu napredovanja najprej ponudimo vprašanja še neobdelane teme. Učenec mora rešiti v naprej določeno število nalog in njegova uspešnost mora biti nad določenim pragom, da lahko štejemo poglavje kot osvojeno. Sledi akcija zamenjava teme, kjer se prej opisani postopek ponovi, kar ponavljamo, dokler ni celoten učni načrt izpopolnjen. Po končanem napredovanju, sledi stopnja ponavljanja. Ker imamo sedaj model učenca že inicializiran in nastavljen, prikazujemo tista poglavja, ki imajo najnižjo uspešnost (vgrajeno je tudi varovalo, ki poskrbi da se pod določeno mejo prikazuejo tudi poglavja z najmanjšim številom ponovitev).

Vsa stanja so trajno hranjena in vplivajo na nadaljnje odločitve. To pomeni, da tudi, ko se učenec vrne k reševanju, bodo njegove pretekle odločitve vplivale na v nadaljevanju prikazane naloge.

3.2 Zaslonske maske

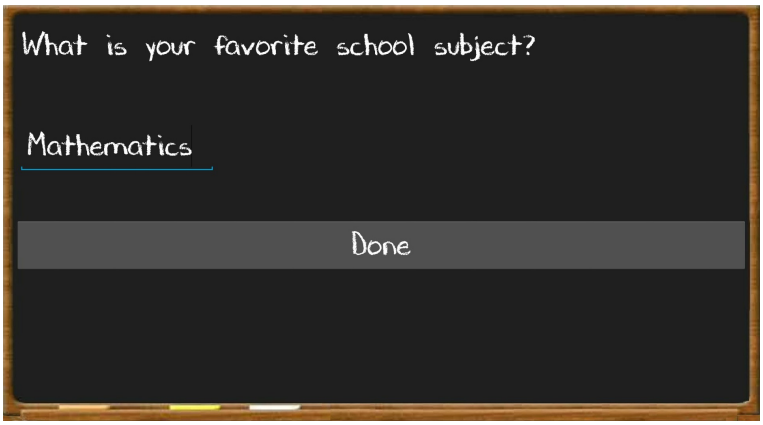
Aplikacija je sestavljena iz štirih zaslonskih mask: glavnega menija, vprašalnika, pregleda profila in maske za reševanje ulomkov. Zadnje tri maske so dosegljive iz glavnega menija, slika 3.1.



Slika 3.1: Glavni meni

3.2.1 Vprašalnik

Vprašalnik je sestavljen iz 59 različnih vprašanj, od tega jih je pet statičnega tipa (slika 3.2), preostala pa so del (MSLQ) [21] vprašalnika, ki nam služi za zajem učnega stila (slika 3.3). Statični stil sprašuje po imenu in priimku, spolu, starosti, najljubšem predmetu in učnem uspehu. Sledi 55 vprašanj izbirnega tipa.



Slika 3.2: Statično vprašanje o najljubšem predmetu.

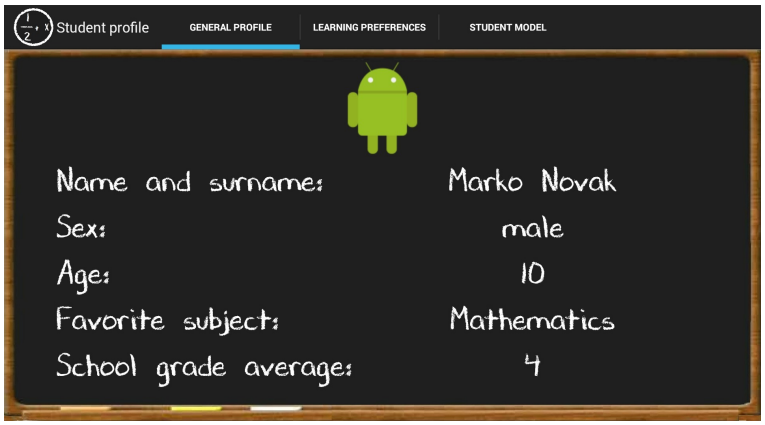
Slika 3.3: Vprašanje v sklopu MSLQ [21] kviza

3.2.2 Profil študenta

Ta zaslonska maska je zgrajena iz več tematskih zavihkov, kar se preslika na Androidne fragmente. Zavihki so trije: splošni profil, učne preference in model učenca.

Splošni profil

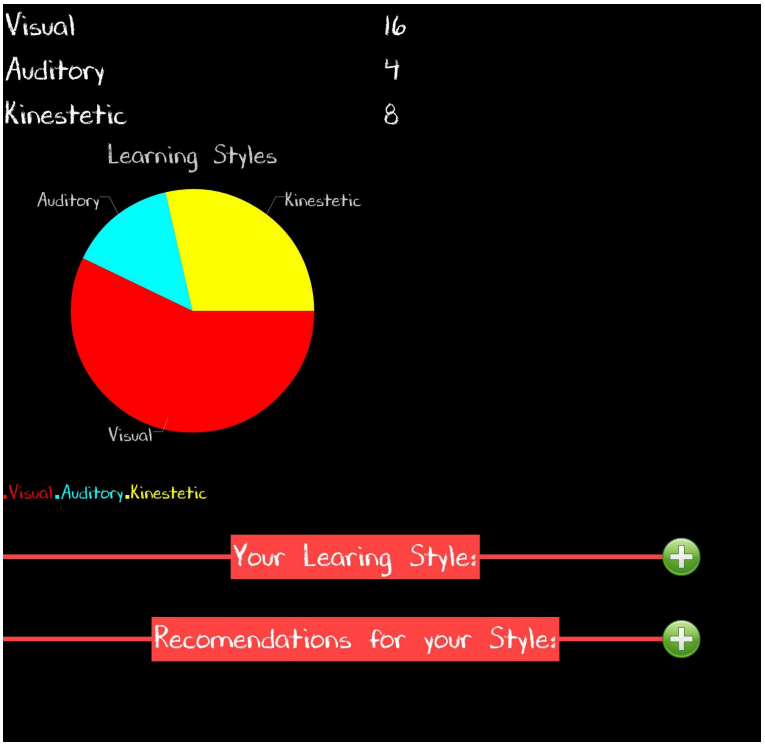
Zavihek splošni profil: slika 3.4 prikazuje splošne podatke, ki smo jih zajeli s pomočjo prej opisanega kviza. Ti podatki so statični, spreminjanje je mogoče samo s ponovnim reševanjem kviza.



Slika 3.4: Prikaz prvega zavihka: slika na vrhu se spreminja glede na spol.

Učne preference

Zavihek učne preference: slika 3.5 pravtako vsebuje podatke iz kviza, ki pa že so v agregirani obliki. Na pogledu se nahajajo kvantificirani rezultati (MSLQ) [21] vprašalnika (število odgovorov, ki so značilna za vsak učni stil), krožni diagram, ki prikazuje deleže posameznih učnih stilov, in dve polji, ki nam podata obsežnejšo razlago in priporočila za dominanten učni stil. Prav tako so tudi ti podatki statični in se do ponovnega reševanja kviza ne spreminjajo.



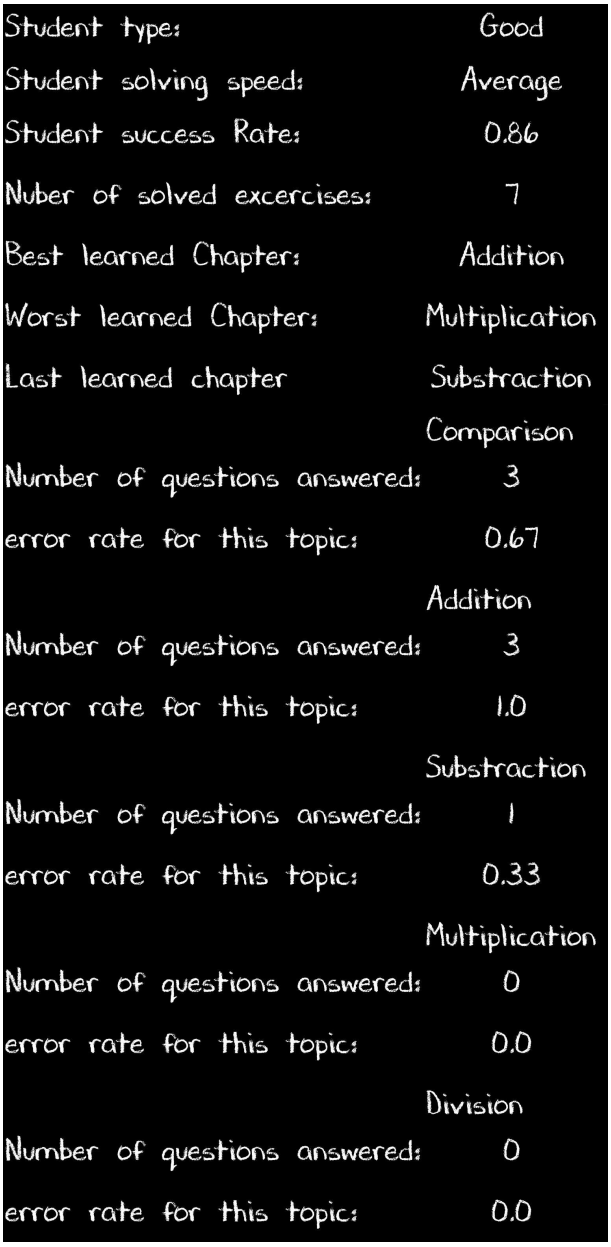
Slika 3.5: Prikaz zavihka učne preference s skritim tekstom razlage in priporočil.

Model učenca

V zadnjem sklopu se nahaja model učenca (slika 3.6), ki vsebuje dinamične podatke. Ti so odvisni od učenčeve interakcije z aktivnostjo Ulomki. Razdelili smo jih na agregirane podatke, ki veljajo za učenca, in na posamezne podatke, ki so vezani na poglavje.

Agregirani podatki so sestavljeni iz tipa učenca (učenca klasificiramo na podlagi povprečnega časa in uspešnosti), povprečnega časa reševanja ene naloge (fuzzy preslikava poskrbi za opisno vrednost), povprečne uspešnosti reševanja (čez vsa poglavja), števila vseh rešenih nalog (čez vsa poglavja), najboljše rešenega poglavja (poglavje z najvišjo uspešnostjo reševanja), najslabše rešenega poglavja (poglavje z najnižjo uspešnostjo reševanja) in trenutno zadnje naučenega poglavja.

Podatki po poglavjih so sestavljeni iz imena teme (primerjava ulomkov, seštevanje, odštevanje, množenje in deljenje), števila reševanj te teme in uspešnosti te teme.

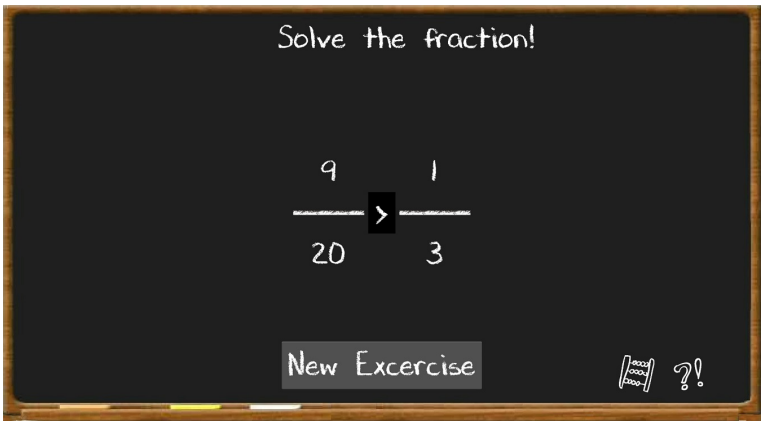


Slika 3.6: Prikaz zavihka modela učenca, ki vsebuje vse dinamične parametre.

3.2.3 Ulomki

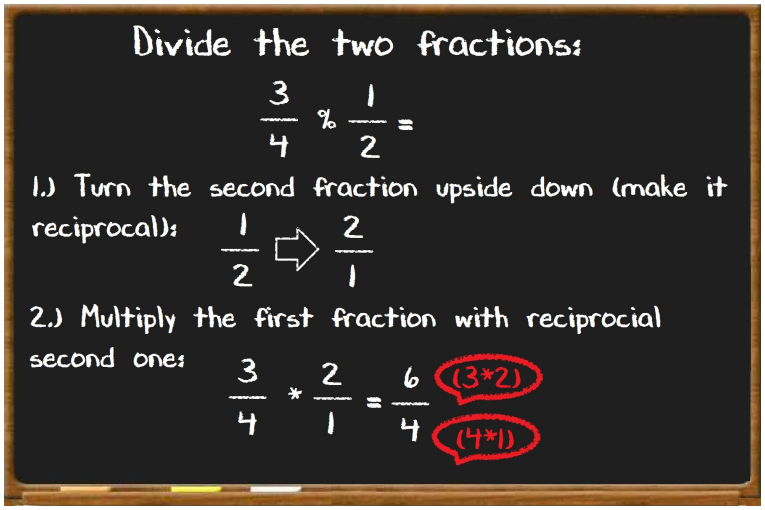
Zaslon ulomki je namenjen prikazovanju in reševanju nalog. Je tudi del, kjer se zbirajo dinamični podatki, ki vplivajo na prikazane naloge. Na zaslonu, kot

prikazuje slika 3.7, imamo poleg naloge in gumba še dve pomoči.



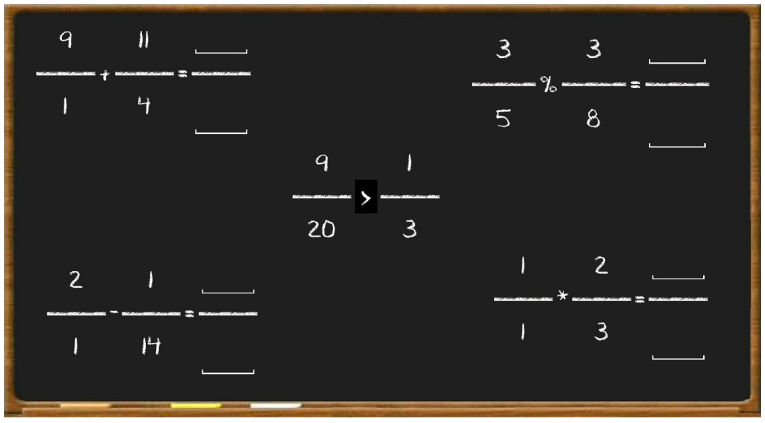
Slika 3.7: Prikaz aktivnosti ulomki, z nalogo primerjanja.

Prva pomoč, ikona abakus, prikaže skupni imenovalec obeh ulomkov, kar v nekaterih primerih znatno olajša reševanje. Druga pomoč, ki se nahaja pod ikono vprašaja in klicaja, pa se dinamično spreminja, glede na učenčev učni stil. Pripravljeni sta dve različici, prva je za učence z auditornim učnim stilom, ki vsebuje pet različnih zvočnih posnetkov, za vsako temo svojega. Vsak posnetek razlaga postopek reševanja različnih nalog. Tako pri seštevanju izvemo vse potrebne korake, da pridemo do rešitve, vključno s primerom. Hkrati pa se prikažeta dva gumba za nadzor zvoka, "play"in "pause". Druga različica, ki združuje vizualne in kinestetične učne stile (zaradi specifičnosti niso mogoči kinestetični namigi), pa pomoč prikaže v obliki slikovnega gradiva (slika 3.8) in navodil. Navodilo je sestavljeno iz petih različnih grafičnih prikazov, ki na primeru ponazorijo potrebne korake za reševanje dotične teme.



Slika 3.8: Prikaz pomoči za poglavje deljenje

Poleg pravilnosti rešitve se zajema tudi čas reševanja posamezne naloge. Prikaz nalog ima pet različnih tematik (slika 3.9) in sicer: primerjava ulomkov, seštevanje, odštevanje, množenje in deljenje. Prvi način delovanja je, da tematike prikazujejo po vrsti, dokler učenec pravilno ne reši minimalnega števila zahtevanih nalog (zdajšnje poglavje se ne spremeni, dokler učenec ne reši pravilno treh nalog v trenutnem poglavju).



Slika 3.9: Prikaz vseh različnih poglavij.

Ko učenec reši ustrezno število nalog, se poglavje spremeni. Ko predelamo,

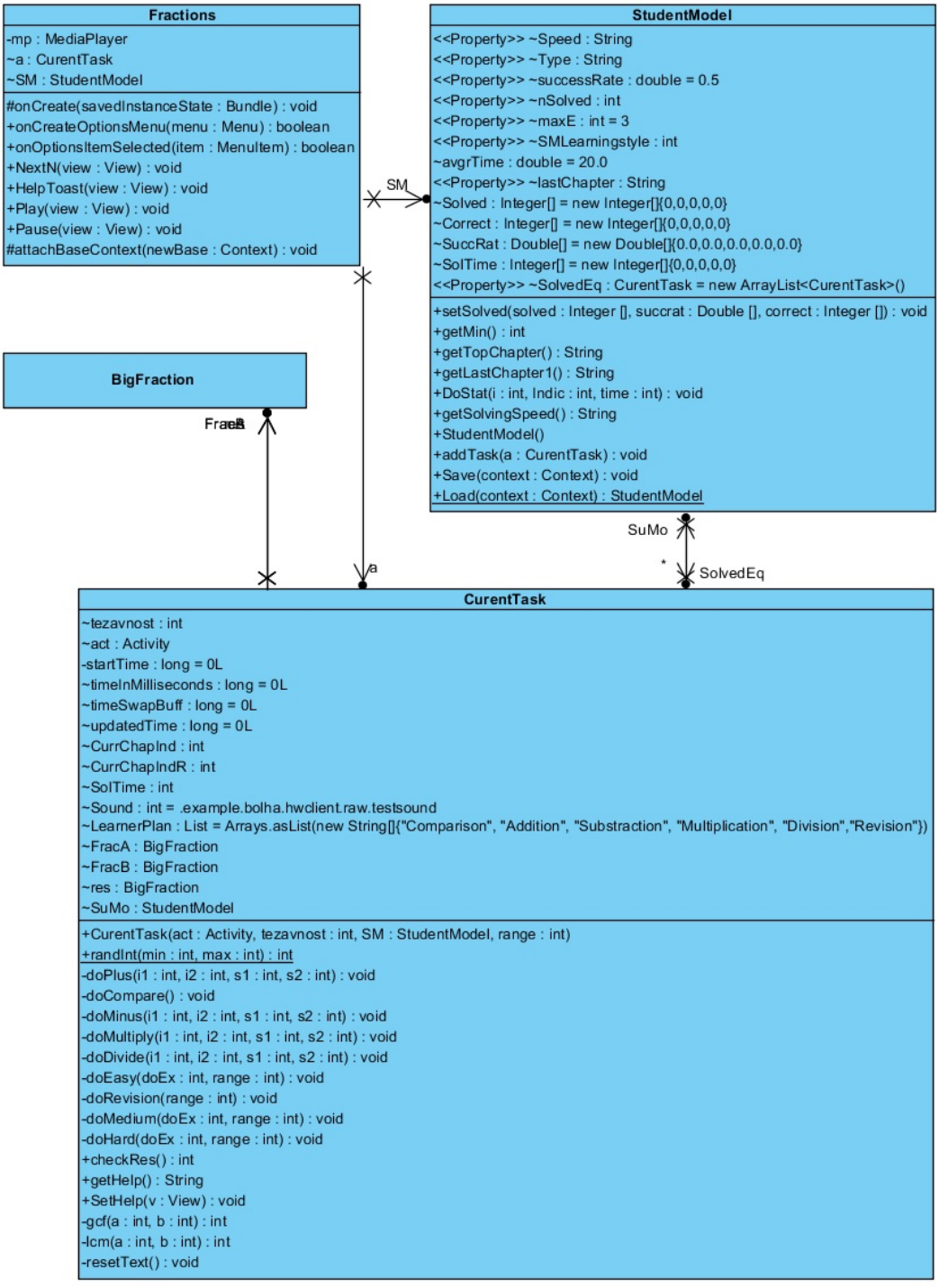
vsa poglavja, vstopimo v način ponavljanja. Izberemo poglavje z najslabšim rezultatom oziroma, da se ne bi pretirano ponavljala najslabša poglavja, najredkeje ponavljano poglavje. V tem načinu, se spremeni tudi težavnost prikazanih nalog.

Naloge imajo spremenljivke tudi med operandi tako, da je potrebno uporabiti tudi obratno logiko. Nabor oziroma vrednost operatorjev se dinamično spreminja. Če bo nekdo hitreje reševal naloge, bo lahko dobil operatorje v vrednosti od ena do petnajst, medtem ko nekdo, ki počasneje rešuje naloge le v vrednosti od ena do deset itd. Vse krmiljenje in nekatere preslikave uporabljajo metode mehkega računanja. Ulomki se lahko vnašajo tudi v neskrajšani obliki.

3.2.4 Inteligentni tutorski sistem

Aplikacija ima tudi druge elemente ITSja, prikazani so na sliki 3.3. Imamo pedagoški modul (Fractions), bazo znanja (BigFraction in CurrentTask) in model učenca (StudentModel). Pedagoški modul s pomočjo stanj modela učenca generira nov objekt tipa CurrentTask, ki prikaže primerno nalogo na aktivnosti ulomki. Razred StudentModel je bil primarno načrtovan za hranjenje stanj, zato vsebuje večinoma "getterje", šetterje"in agregatorske funkcije. Razred Fractions, ki skrbi za krmiljenje, določa zamenjavo naloge in izračuna časovno težavnost. Razred CurrentTask skrbi za izbiro poglavja, težavnosti in prikaz naloge. Podatki se trajno hranijo s pomočjo Shared Preferences in zapisovanja v datoteke.

Projekt vsebuje okoli 4000 vrstic javanske kode, najdaljši razred CurentTask.java sestavlja 1186 vrstic kode. Skupno število vrstic z oblikovanjem in konfiguracijo se povzpne na okoli 12500 vrstic.



Slika 3.10: Pedagoški modul, baza znanja in model učenca.

3.3 Model mehke logike

Z modelom mehke logike smo realizirali pedagoški modul in dele v modelu učenca. V pedagoškem modulu smo uporabili tri različne modele in sicer za ugotavljanje učenčeve uspešnosti, hitrosti in za izvajanje akcij pri izbiranju nove naloge. V modelu učenca pa model mehke logike uporabljamo za klasifikacijo učenca in povprečne hitrosti reševanja. Poglavitna lastnost, zaradi katere smo se odločili za takšno orodje, je nedeterminističnost. Kot smo ugotovili že v teoriji, se takšne realizacije obnesejo bolje.

Vsi modeli uporabljajo Mamdani kontroler, pripadnostne funkcije so trikotne oblike, uporablja se min max metoda in ostrenje. Odmehčanje ("defuzzyfikacija") je integralski centroid (izračun težišča dobljenega lika).

```
Engine engine = new Engine();
engine.setName("StudenType");

InputVariable inputVariable = new InputVariable();
inputVariable.setEnabled(true);
inputVariable.setName("AverageSolSpe");
inputVariable.setRange(0.000, 100.001);
inputVariable.addTerm(new Triangle("FAST", 0.000, 10.00, 14.000));
inputVariable.addTerm(new Triangle("MEDIUM",12.250, 20.500,
    25.750));
inputVariable.addTerm(new Triangle("SLOW", 20.500, 35.00,
    100.001));
engine.addInputVariable(inputVariable);

InputVariable inputVariable2 = new InputVariable();
inputVariable2.setEnabled(true);
inputVariable2.setName("AverageSolSuccess");
inputVariable2.setRange(0.000, 1.000);
inputVariable2.addTerm(new Triangle("POOR", 0.000, 0.3500,
    0.500));
```

```
inputVariable2.addTerm(new Triangle("MEDIUM",0.450, 0.65, 0.750));
inputVariable2.addTerm(new Triangle("GOOD", 0.650, 0.750, 0.82));
inputVariable2.addTerm(new Triangle("VERYGOOD", 0.750, 0.850,
    1.00));
engine.addInputVariable(inputVariable2);

OutputVariable outputVariable = new OutputVariable();
outputVariable.setEnabled(true);
outputVariable.setName("SType");
outputVariable.setRange(0.000,1.00);
outputVariable.fuzzyOutput().setAccumulation(new Maximum());
outputVariable.setDefuzzifier(new Centroid(200));
outputVariable.setDefaultValue(0.65);
outputVariable.setLockValidOutput(false);
outputVariable.setLockOutputRange(false);
outputVariable.addTerm(new Triangle("POOR1", 0.000, 0.3500,
    0.500));
outputVariable.addTerm(new Triangle("MEDIUM1",0.450, 0.65,
    0.750));
outputVariable.addTerm(new Triangle("GOOD1", 0.650, 0.750, 0.82));
outputVariable.addTerm(new Triangle("VERYGOOD1", 0.750, 0.850,
    1.00));
engine.addOutputVariable(outputVariable);

RuleBlock ruleBlock = new RuleBlock();
ruleBlock.setEnabled(true);
ruleBlock.setName("");
ruleBlock.setConjunction(new Minimum());
ruleBlock.setDisjunction(new Maximum());
ruleBlock.setActivation(new Minimum());
ruleBlock.addRule(Rule.parse("if AverageSolSpe is FAST and
    AverageSolSuccess is VERYGOOD then SType is VERYGOOD1",
    engine));
```

```
ruleBlock.addRule(Rule.parse("if AverageSolSpe is FAST and
    AverageSolSuccess is GOOD then SType is GOOD1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is FAST and
    AverageSolSuccess is MEDIUM then SType is MEDIUM1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is FAST and
    AverageSolSuccess is POOR then SType is POOR1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is MEDIUM and
    AverageSolSuccess is VERYGOOD then SType is GOOD1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is MEDIUM and
    AverageSolSuccess is GOOD then SType is MEDIUM1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is MEDIUM and
    AverageSolSuccess is MEDIUM then SType is MEDIUM1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is MEDIUM and
    AverageSolSuccess is POOR then SType is POOR1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is SLOW and
    AverageSolSuccess is VERYGOOD then SType is MEDIUM1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is SLOW and
    AverageSolSuccess is GOOD then SType is MEDIUM1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is SLOW and
    AverageSolSuccess is MEDIUM then SType is POOR1", engine));
ruleBlock.addRule(Rule.parse("if AverageSolSpe is SLOW and
    AverageSolSuccess is POOR then SType is POOR1", engine));
engine.addRuleBlock(ruleBlock);

StringBuilder status = new StringBuilder();
if (!engine.isReady(status))
throw new RuntimeException("Engine not ready. " +
"The following errors were encountered:\n" + status.toString());
```

Listing 3.1: Model mehke logike StudenType, realiziran s knjižnico jfuzzy, kot vhod mu podamo povprečno hitrost reševanja in povprečno uspešnost.

Za realizacijo modela smo uporabili prostodostopno knjižnico jfuzzylite. Izgradnja se začne z inicializacijo vhodnih in izhodnih spremenljivk ter bloka pravil. Ko imamo model postavljen z dodeljevanjem vrednosti vhodnim sprememl jivkam krmilimo izhodne. Zgornji prikaz se nanaša na model StudenType, kjer hočemo klasificirati učenca na slabega, povprečnega, dobrega in zelo dobrega. Prvi korak inicializira model in mu določi enolično ime. Sledi definicija prve vhodne spremenljivke AverageSolSpe, ta nam predstavlja povprečni reševalni čas na posamezno nalogo. Razpon te spremenljivke smo omejili od 0 do 100.001 sekundo, sledi razdelitev vhodne spremenljivke na tri vrednosti, vsaka s svojo veljavnostjo. Hitrost smo razdelili na počasno, srednje in hitro. Druga spremenljivka se imenuje AverageSolSuccess in nam predstavlja povprečen uspeh reševanja posamezne naloge. Vrednost vhoda je omejena na vrednosti med nič in ena, spremenljivka se deli na štiri vrednosti slabo, srednje, dobro in zelo dobro. Definicija izhodne spremenljivke SType je omejena na vrednosti med nič in ena, privzeto vrednost smo nastavili na povprečno. Izhodna vrednost je pravtako razdeljena na štiri dele kot vhodna spremenljivka dva. Sledi nastavljanje pravil, ki nam povežejo vhodne spremenljivke z izhodnimi. Na podlagi teh pravil se izračuna funkcija. Ta model nam na podlagi povprečnega časa reševanja da oceno učenca. Ocena učenca se "popravi", glede na povprečen čas reševanja. To pomeni, da počasen učenec ne more nikoli biti želo dober učenec", čeprav njegova uspešnost to nakazuje.

Poglavje 4

Uporabljene tehnologije in orodja

4.1 Metode razvoja

Pri razvoju so bile uporabljene agilne metodologije [18], ki predpostavljajo, da je delujoča programska oprema pomembnejša kot popolna dokumentacija. Tako smo izdelali prototip z vsemi zahtevanimi funkcionalnostmi, vendar nepopolno dokumentacijo. Komunikacija je bila zelo olajšana, saj je bila vloga naročnika, razvijalca in vodje projekta združena. Smer razvoja je sledila začetnemu načrtu z nekaj manjšimi dodelavami, vendar to ni vplivalo na odzivnost in prilagodlivost na spremembe. Razvojna strategija je bila iterativni razvoj, odlašanje izvedbe za določeno časovno obdobje z namenom hitrejšega napredovanja in kasnejšega vračanja k istemu problemu z namenom dokončanja. Različne iteracije so nam predstavljale funkcionalnosti, ki smo jih najprej realizirali v omejenem obsegu z namenom testiranja koncepta pred dokončno realizacijo.

4.2 Android

Android je na Linuxu temelječ mobilni operacijski sistem, največ razvoja na tem področju pade pod okrilje podjetja Google. Ocenjuje se, da je trenutno na svetu več kot 1,4 milijarde naprav, ki jih poganja ta operacijski sistem. Poleg tabličnih

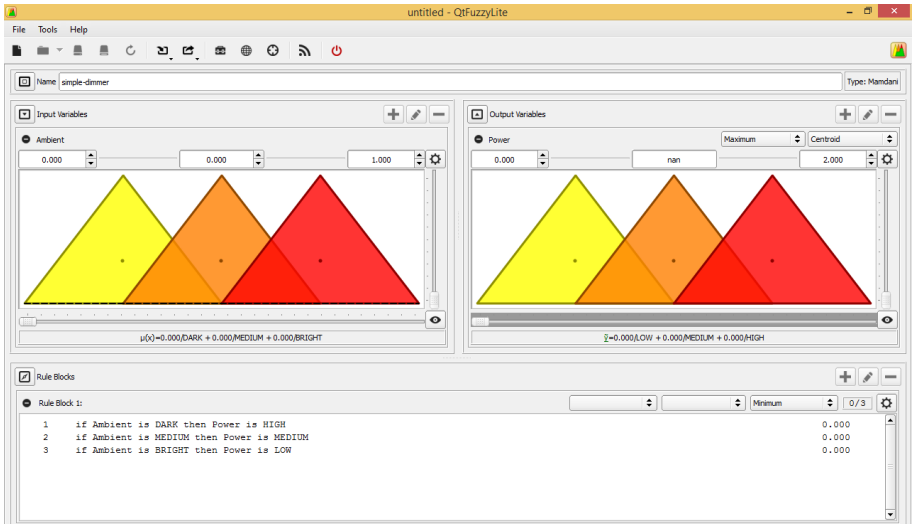
računalnikov in mobilnih telefonov se platforma širi tudi na pametne ure, televizije in avtomobile, kar kaže na veliko težnjo poenotenega okolja oziroma "the internet of things". Integracija, deljenje in zajemanje različnih podatkov zelo olajša in omogoča nove priložnosti.

4.3 Android Studio

Je namensko razvojno okolje za sistem Android, za katerim stoji podjetje Google. Najavljeno je bilo 16. maja 2013, do junija 2014 pa je bilo na voljo samo v zgodnjem predogledu. Razvoj se je začel na prvi beta verziji z oznako 0.8, decembra 2014 pa se je nadgradil na verzijo 1.0. Samo orodje je še novo in kaže Googlevo težnjo k čim lažjemu širjenu Androida, saj vsebuje podporo za vse izpeljanke, od telefonov in tablic do pametnih zapestnic in ur na enem mestu. Pred stabilno verzijo Android Studio se je za razvoj po večini uporabljal modificiran Eclipse, na katerega je bilo potrebno dodajati razširitve. V času razvoja, kljub zgodnji beta verziji, razen na občasno počasnost nisem naletel na kakšne večje težave.

4.4 Knjižnica fuzzylite

Fuzzylite je brezplačna in odprtokodna knjižnica za mehko računanje, napisana za več platform. Namenjena je predvsem za izdelavo kontrolerjev z uporabo mehke logike, pri tem pa ne potrebujemo kakšne dodatne knjižnice. Knjižnica je trenutno na voljo za C++ in Javo. Omogoča nam objektno izgradnjo modela mehke logike. Takšen model potem na podlagi vhodnih spremenljivk s pomočjo mehke logike tvori izhodne spremenljivke. Vsebuje več različnih kontrolerjev, lingvističnih izrazov t-normov itd ..., kar omogoča veliko svobode pri oblikovanju modelov. Plačljiv dodatek Qtfuzzylite na sliki 4.1 nam omogoča grafično izgradnjo in test modela, ki ga lahko izvozimo v več različnih formatov.



Slika 4.1: Plačljivi dodatek Qtfuzzylite. Na sredini se nahajajo vhodne in izhodne spremenljivke, spodaj pa lahko vidimo pripadajoče terme.

4.5 Knjižnica Generic Quest Library for Android [19]

Generic Quest Library for Android (GQL8) je knjižnica, ki nam omogoča lažjo implementacijo vprašalnika. Vprašanja podamo kot dokument JSON in tudi rezultate dobimo v obliki dokumenta JSON. Knjižnica omogoča pet različnih načinov vnosa podatkov. Pri implementaciji aplikacije se je pojavila potreba po prostovnosnem polju "textbox", ki ga osnovna implementacija ne omogoča, zato je bila potrebna dodelava osnovne knjižnice. Dodano je bilo še trajno hranjenje rezultatov.

4.6 Knjižnica AChartEngine [23]

AChartEngine knjižnica je namenjena enostavnemu prikazu grafov v Android okolju. Uporablja nativne elemente, s čimer odpade potreba po dodatnih razširitvah. Odlikuje jo tudi enostavna uporaba, ki nam omogoča bogat nabor urejanja tako izgleda kot funkcionalnosti.

4.7 Knjižnica BigFraction [24]

BigFraction je odprtokodna javanska knjižnica, ki omogoča osnovne operacije z ulomki. Avtor jo je naredil za reševanje problemov na strani Project Euler. Odlikuje jo enostavna implementacija in uporaba.

Poglavje 5

Sklepne ugotovitve

Cilj naloge je bil proučiti modele učenca v inteligentnih tutorskih sistemih tako teoretično kot praktično. Za teoretično podlago smo vzeli pregled literature zadnjih desetih let [1]. Za bolj uravnoteženo in poglobljeno analizo smo dodatno proučili še članke od [1] do [17]. S tem smo si zagotovili objektivnejši pregled področja in zadostili začetnim zahtevam. Tematika ima nekaj glavnih področij razvoja, pri čemer so nekatera bolj perspektivna kot druga. Kar še bode v oči, je omejeno število člankov na tem področju.

Ključni cilj praktičnega dela diplomskega dela je bil zasnova in realizacija aplikacije za operacijski sistem Android. Načrtovanje se je začelo z izbiro tipa modela učenca in učne domene, kjer smo izbrali model mehke logike in stereotipnega, kasneje so bili dodani elementi prekrivnega modela. Realizirali smo celoten ITS z vsemi osnovnimi komponentami in delujočimi funkcionalnostmi.

Glede na to da so glavni viri teoretičnega dela diplomskega dela znanstveni članki, je njegov glavni prispevek začetna podlaga za nadaljnjo raziskovanje področja. Vtis med začetno analizo je bil, da je področje zaenkrat še omejeno in da objave niso raznovrstne, raziskovalno področje je kompleksno in včasih zahteva interdisciplinarnost. Diplomsko delo lahko služi tudi kot pregled znanstvene literature. Prispevki praktičnega dela diplomskega dela so implementacija inteligentnega tutorskega sistema na platformi Android, proučitev in integracija knjižnice za mehko logiko jFuzzyLogic in implementacija modela učenca z uporabo mehke logike.

Prispevki teoretičnega dela so predvsem akademske narave, služijo lahko kot

izhodišče za nadaljnje raziskovanje ali pa kot nekakšen hiter povzetek področja. Prav tako lahko služi kot teoretična podlaga za razvoj podobne aplikacije, kot smo jo razvili mi. Vsako predstavljeno poglavje vsebuje seznam že razvitih sistemov.

Razvita aplikacija lahko služi kot prototip, oziroma kot test koncepta. Med proučenimi viri še ni bilo veliko takšnih, ki bi poskušali implementirati ITS na mobilne platforme. Med realizacijo smo tudi preizkušali delovanje knjižnice jFuzzyLogic na Androidu. Z zasnovano aplikacijo smo dosegli vse začrtane cilje in proučili nekatera orodja, ki omogočajo takšno implementacijo. Aplikacija lahko služi kot praktični prikaz ITS-ja v učnem procesu. Razvita aplikacija se zaenkrat ne uporablja, vse funkcionalnosti pa so razvite in pripravljene za uporabo.

Rezultat diplomskega dela je celovit pregled objav področja različnih implementacij modela učenca in delujoča aplikacija, ki je namenjena predvsem demonstraciji. Prostor za izboljšavo je predvsem v kompleksnosti modelov mehke logike, kjer bi bilo potrebno analizirati tudi druge dostopne metode implementacije in različne modele. Prav tako bi se dalo razširit učno domeno na druga matematična področja in na vprašanja izbirnega tipa. Implementiran je tudi komunikacijski kanal na strežnik, vendar je trenutno neaktiven, kar odpira vrata za veliko različnih funkcionalnosti. Morda bi bilo smiselno razmisliti tudi o implementaciji "bug-library", ki v naši aplikaciji še ni definiran.

Literatura

[1] Konstantina Chrysafiadi, Maria Virvou, "Student modeling approaches: A literature review for the last decade", Expert Systems with Applications: An International Journal archive, Volume 40 Issue 11, September, 2013

[2] Yujian Zhou, Martha W. Evens, "A Practical Student Model in an Intelligent Tutoring System", Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on, 1999

[3] Dongming Xu, Huaiqing Wang and Kaile Su, "Intelligent Student Profiling with Fuzzy Models", System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on, 2002

[4] Andrej Marolt, Jernej Mlakar, "Inteligentni tutorski sistemi", Seminar: Umetni inteligentni sistemi

[5] Alenka Kavčič, Rafael Pedraza-JimBnez, Harold Molina-Bulla, Francisco J. Valverde-Albacete, Jesus Cid-Sueiro, and Angel Navia-Vazquez, "Student Modelling Based on Fuzzy Inference Mechanisms", EUROCON 2003. Computer as a Tool. The IEEE Region 8 (Volume:2)

[6] R. Stathacopoulou, G. D. Magoulas, M. Grigoriadou, "Neural Network-based Fuzzy Modeling of the Student in Intelligent Tutoring Systems", Neural Networks, 1999. IJCNN '99. International Joint Conference on (Volume:5)

[7] Slika 2.3 dostopna na:
`http://www.atp.ruhr-uni-bochum.de/rt1/syscontrol/node123.html`

[8] R. Antonija Mitrovic, Stellan Ohlsson, "Evaluation of a Constraint-Based Tutor for a Database Language", International Journal on Artificial Intelligence in Education 1999, 10(3–4), pp. 238–56.

[9] Brent Martin, "Constraint-Based Modeling: Representing Student Knowledge", New Zealand Journal of Computing 1999 volume 7

[10] Jay Holland, Antonija Mitrovic, Brent Martin, "J-LATTE: a Constraint-based Tutor for Java", Proceedings of the 17th International Conference on Computers in Education 2009

[11] Julia Clemente, Jaime Ramírez, Angélica de Antonio , "A proposal for student modeling based on ontologies and diagnosis rules", Expert Systems with Applications Volume 38, Issue 7, July 2011, Pages 8066–8078

[12] Desislava Paneva, "Ontology-based Student Modeling", CHIRON Open Workshop "Ubiquitous Learning Challenges: Design, Experiments and Context Aware Ubiquitous Learning"; 09/2006.

[13] Weiqin Chen, Yusuke Hayashi, Lai Jin, Mitsuru Ikeda, Riichiro Mizoguchi, "An Ontology-based Intelligent Authoring Tool", In Proc. of the Sixth International Conference on Computers in Education 1998

[14] Ortony, A. et al., "The cognitive structure of emotions", Cambridge Univ. Press, UK, 1988.

[15] Patricia A. Jaques, Rosa M. Viccari, "A BDI Approach to Infer Student's Emotions",Computers and Education Volume 49, Issue 2, September 2007, Pages 360–384.

[16] Hao Cen, Kenneth Koedinger, and Brian Junker, "Learning Factors Analysis – A General Method for Cognitive Model Evaluation and Improvement",Intelligent Tutoring Systems Lecture Notes in Computer Science Volume 4053, 2006, pp 164-175.

[17] Karla Muñoz , Julieta Noguez , Paul Mc Kevitt, Luis Neri , Víctor Robledo-Rella, and Tom Lunney, "Adding Features of Educational Games for Teaching Physics",Frontiers in Education Conference, 2009. FIE '09. 39th IEEE

[18] Marko Bajec, Marjan Krisper, "AGILNE METODOLOGIJE RAZVOJA INFORMACIJSKIH SISTEMOV ", Univerza v Ljubljani, 01/2003

[19] Knjižnica Generic Quest Library for Android dostopna na:
`http://www.catedrasaes.org/wiki/CarimQuestionnaires`

[20] slika3 dostopna na:
`http://www.ruebenstrunk.de/emeocomp/4e.HTML`

[21] MLSQ dostopna na:
`https://downloads.newcastle.edu.au/divisions/academic/centre%20for%20teaching%20and%20learning/academic-development/mlsq%20interpretation%20information.pdf` , dostopno 4.12.2014

[22] Juan Rada-Vilela, "fuzzylite: a fuzzy logic control library ", 2014. dostopna na: `http://www.fuzzylite.com`

[23] achartengine dostopna na:
`http://www.achartengine.org/`

[24] bigfraction dostopna na:
`https://github.com/kiprobinson/BigFraction`